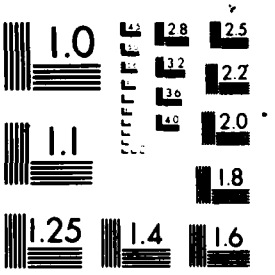
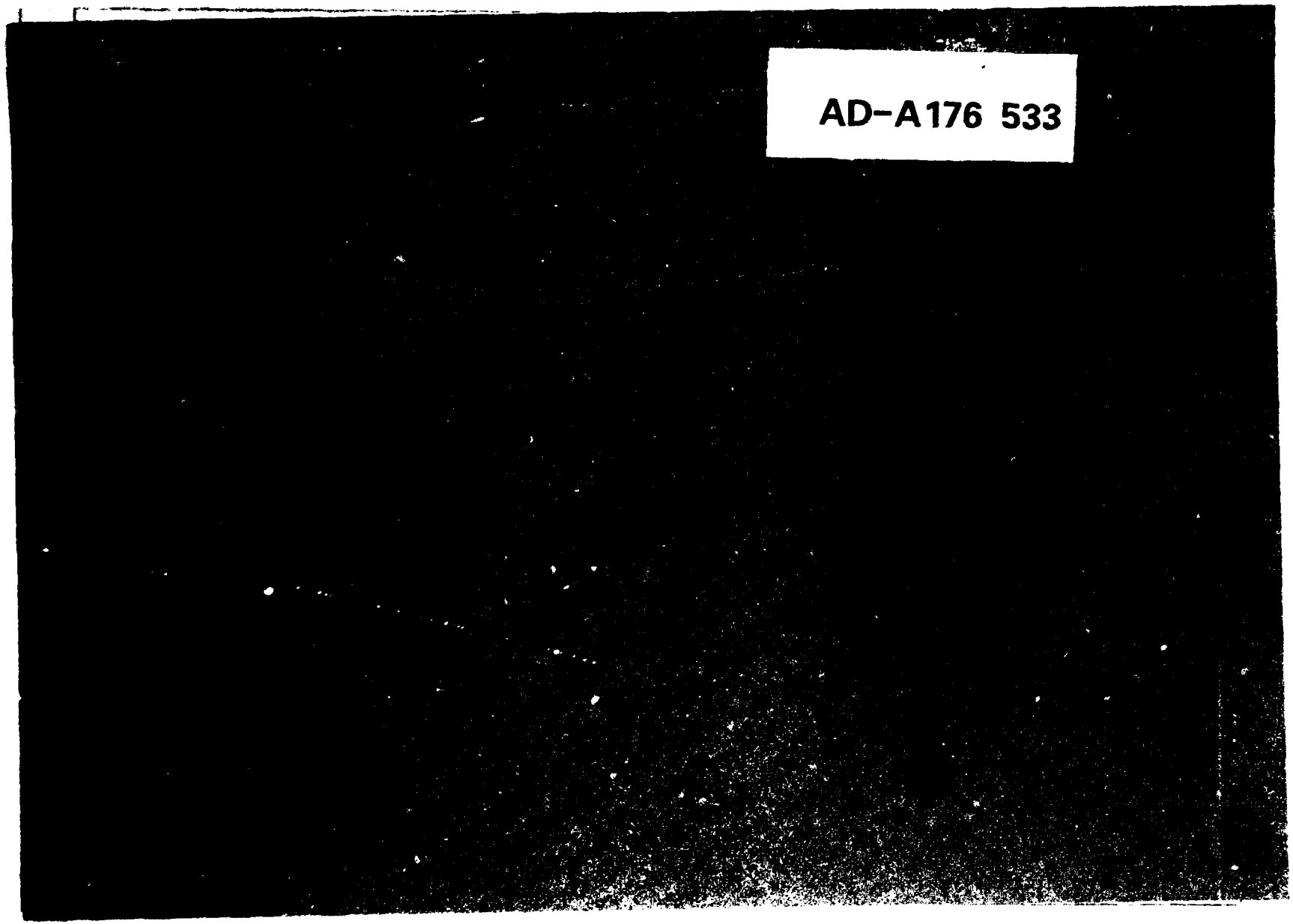
AD-A176 533 TABULATED OUTPUT FROM ELIA(U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) K R MILNER AUG 86 RSRE-MEMO-3966 DRIC-BR-101108 1/1

UNCLASSIFIED F/G 9/2 NL

END




AD-A176 533

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3966

TITLE: TABULATED OUTPUT FROM ELLA

AUTHOR: K R Milner

DATE: August 1986

SUMMARY

An addition to the ELLA system is described, which takes the information stored by the simulator in a history file and outputs it in tabular form. Monitors are displayed horizontally and their values are tabulated against time, which runs vertically.

RSRE MEMORANDUM NO 3699

TITLE: TABULATED OUTPUT FROM ELLA

AUTHOR: K R Milner

CONTENTS

1. INTRODUCTION

ELLA is a Hardware Design and Description Language that has been developed at RSRE over the past eight years. It allows the user to form a description of a circuit (at different levels of abstraction) and then to simulate his description. If there is a discrepancy between the simulator output and the expected output, then he can alter the ELLA description of his hardware model and re-simulate. It is therefore important that the simulator has a good user-interface: in particular, that the output is in a form which can be easily read and compared with the expected output. This memo describes a form of output - tabular output - which is an alternative to the current output.

The interactive tabular might be preferred to the existing simulator for analysing ELLA output for four main reasons. Firstly, when a number of different points in the circuit are being monitored simultaneously, it is easier to follow the changes of one particular monitor point, since the different values it takes up are all placed in the same column. Secondly, if a monitor point is a complex ELLA structure, consisting of a number of different fields, then it is possible to select individual elements of the structure, whereas in the current simulator, all fields would have to be displayed. Thirdly, there is a facility for searching for all occurrences of an event or combination of events and finally, there is the possibility of renaming basic values in the tabulator; this can be used to generate simple waveforms as described in Section 3.4.

The two different output formats are compared in the appendix, in which the same information is shown in the output format of the current simulator (APPENDIX 1) and in tabular form (APPENDIX 2).

The tabulator is written in ALGOL 68 (like the rest of the ELLA system) and uses a syntax manipulator called SID[1]; this enables new commands to be added and existing ones to be altered with the minimum of effort. It was originally written as a post-processor quite separate from the rest of the ELLA system, whereas now it is being fully integrated as an interactive facility. This report describes the facilities of the post-processor because the definition of the interactive system has not yet stabilised. The differences between the two are expected to be of a mainly cosmetic nature though; the current version of the interactive system is described in Chapter 17 of the ELLA User Manual [2].

2. SIMULATOR COMMANDS

The tabulator uses data stored by the simulator in a history file; this includes information about the ELLA TYPEs used and changes in value of the monitor points that have been kept. The simulator commands needed to produce a history file which is to be used by the tabulator are described below.

2.1 History

History enables the user to name the history file which is created by the simulator; the command

```
history histfile
```

produces the history file histfile. If no history command is given, a default filename is used: for example, on VAX, if the user's ELLA library is zzzlib.ell, then the default history file will be zzzlilb.elh.

2.2 Tabulate

Tabulate is used to enter the tabulator. For example

```
tabulate histfile
```

will enter the tabulator with the history file histfile. If the tabulate command is given from the simulator and no filename is given, the tabulator will use the history file that has been opened by the simulator.

2.3 mk

mk (monitor keep) is used to name the monitors to be used in the tabulator. The command has the same form as the mc command and like mc, only changes in value of the named monitors are recorded.

3. TABULATOR COMMANDS

The tabulator is an interactive system, which sends output to the screen by default. Output is produced by a window command (see Section 3.3), which displays all monitor points which have been given as parameters to a dp command (Section 3.1). A number of monitor points can be specified by a single name (see Section 3.2). Individual monitor points unfortunately cannot be deleted at the moment; there is only a command to delete all existing monitors - clear display (see Section 3.2).

3.1 dp

dp (display) followed by one or more monitors adds those monitors to the list of monitors currently being displayed.

Monitors can be indexed in a similar way to the indexing of ELLA structures. For example, if mon1 is a structure identifier of TYPE [8] bool, then

```
dp mon1 [1]
```

will display the first element of the [8] bool structure,

```
dp mon1 [1..4]
```

will display the first four elements of mon1 and

```
dp mon1 [1..6,8]
```

will display all elements of mon1 except the seventh.

The number of elements in each monitor being output is indicated by a series of numbers in the heading of each page. For an example which illustrates the format, see APPENDIX 2: the monitors myram and outval are structures consisting of a single element and the monitors acc and address

are two and four element structures respectively.

Note that in the current version of the tabulator, all structures have been 'flattened'; this means that structures of the form ([4] bool, [4] bool) and ([7] bool, bool) are effectively equivalent to [8] bool and should be indexed as such.

3.2 cd

cd (clear display) clears the list of monitors currently being displayed. There is an optional name parameter: for example, if there are three monitors currently being displayed mon1, mon2 and mon3, then

```
cd disp
```

will clear the display and associate the monitors mon1, mon2 and mon3 with the name disp. If this particular display is required again at some time in the future, then it can be recalled by the command

```
dp disp
```

3.3 wi

wi (window) command triggers the output, in the same way that the command ti produces output in the simulator. The main difference between the two is that in the tabulator it is possible to display values at any time earlier than the current time without having to restart at time zero and reset all the monitors.

wi can be given without parameters and this sends output to the screen over the entire range of the history file. A more selective output can be specified in three ways:

```
wi  ..50
```

will output changes in value of the monitors currently being displayed from the start of the history file until time 50.

```
wi  50..
```

will produce output from time 50 until the end of the history file and

```
wi  50..100
```

will output changes between times 50 and 100.

3.4 rn

The command rn (rename) is used to rename any basic value name. It can be used either to abbreviate excessively long basic values, so that more monitors can fit on each screen width or to generate simple waveforms. For an example of rn which demonstrates the latter use, consider the command

```
rn  f="0  "  x=" X "  t="   1"
```

This will generate a display in which an occurrence of 0 in the first column of the tabulated output represents f, X in the second column represents x and 1 in the third column represents t. The effect of such a rename command on the output can be seen in APPENDIX 3.

Note that the new basic value name is specified within quotes and that spaces are allowed as part of the new name. The only restriction on the new name is that it cannot include quote symbols.

3.5 ev

ev (event) is used to search for the times at which a given event has occurred.

3.5.1 Single events

The simplest type of event command is of the form

```
ev mon1 = t
```

(where mon1 is of TYPE bool and bool can be either t, f or x). All instances of this event will be searched for and an appropriate message will be sent to the screen: either "event not found" if no occurrences of the event have been found, or a message to indicate at what times the event occurred.

The command to search for either of the basic values t or f is

```
ev mon1 = (t|f)
```

The symbol "^" can be used for negation:

```
ev mon1 =^x
```

which incidentally would have the same effect as the previous command. It is also possible to search for the occurrence of question marks.

3.5.2 Multiple events

The syntax for multiple events is similar to the syntax for the ELLA CASE statement. For example, if the monitors mon1, mon2 and mon3 are of TYPE bool, then the command

```
ev  (mon1, mon2, mon3) = (t, t, t)
```

will search for the basic value t at all three monitors. More than one set of possibilities can be tested at once:

```
ev  (mon1, mon2, mon3) = ((t, t,t)|(f, f, f)|(x, x, x))
```

will find the times at which all three monitors simultaneously take up the same value.

Monitors on the lefthandside of an event expression can be indexed as described in Section 3.1.

3.5.3 Associated types

To illustrate the use of the event command with associated types, consider the declarations

```
TYPE bool = NEW (t|f|x),

word = [8] bool,

address = NEW (data & word|instr & word).
```

If the monitor mon1 is of TYPE address, then the command

```
ev (mon1//instr) [1] = t
```

will search for the basic value in the first element if the TYPE word associated with instr.

Note that associated types are recovered on the lefthandside of an event expression; this allows structured types (such as word in the above example) to be indexed as before.

If the monitor mon2 is of type [2] address, then the first or second element of the structure can be selected by indexing before recovering the associated type. For example

```
ev (mon2 [1] //instr) [7] = t
```

will search, in the first element of mon2, for the basic value t in the seventh element of TYPE word associated with instr.

3.6 fi

fi exists the tabulator and re-enters the simulator.

REFERENCES

[1] Foster J M, A Syntax Improving Program, The Computer Journal, Vol 11, No 1, pp 31-34, 1968.

[2] The ELLA User Manual Issue 2.0d, produced by Praxis Systems plc, 1986.

APPENDIX 1

```
  *** TIME = 0 ***
ADDRESS = X X X X, MYRAM = LOAD X, ACC = I/1 J/0, OUTVAL = V/0
  *** TIME = 1 ***
MYRAM := LOAD F
  *** TIME = 2 ***
ADDRESS := F F X X, ACC := I/2 J/0
  *** TIME = 3 ***
ADDRESS := T F X X, MYRAM := LOAD T, ACC := I/2 J/3
  *** TIME = 4 ***
ADDRESS := F F X F
  *** TIME = 5 ***
ADDRESS := X F X F, MYRAM := LOAD F, ACC := I/4 J/3
  *** TIME = 6 ***
ADDRESS := F F X F
  *** TIME = 7 ***
ADDRESS := F F F F, MYRAM := LOAD T, ACC := I/6 J/3
  *** TIME = 8 ***
ADDRESS := F T F F, ACC := I/6 J/6
  *** TIME = 9 ***
MYRAM := READ F
  *** TIME = 10 ***
ADDRESS:= T T F F, ACC := I/8 J/6, OUTVAL := V/25
```

APPENDIX 2

	ADDRESS 1	2	3	4	MYRAM 1	ACC 1	2	OUTVAL 1
0	X	X	X	X	LOAD & X	I/1	J/0	V/0
1	X	X	X	X	LOAD & F	I/1	J/0	V/0
2	F	F	X	X	LOAD & F	I/2	J/0	V/0
3	T	F	X	X	LOAD & T	I/2	J/3	V/0
4	F	F	X	F	LOAD & T	I/2	J/3	V/0
5	X	F	X	F	LOAD & F	I/4	J/3	V/0
6	F	F	X	F	LOAD & F	I/4	J/3	V/0
7	F	F	F	F	LOAD & T	I/6	J/3	V/0
8	F	T	F	F	LOAD & T	I/6	J/6	V/0
9	F	T	F	F	READ & F	I/6	J/6	V/0
10	T	T	F	F	READ & F	I/8	J/6	V/25

APPENDIX 3

	ADDRESS 1	2	3	4	MYRAM 1	ACC 1	2	OUTVAL 1
0	X	X	X	X	LOAD & X	I/1	J/0	V/0
1	X	X	X	X	LOAD & 0	I/1	J/0	V/0
2	0	0	X	X	LOAD & 0	I/2	J/0	V/0
3	1	0	X	X	LOAD & 1	I/2	J/3	V/0
4	0	0	X	0	LOAD & 1	I/2	J/3	V/0
5	X	0	X	0	LOAD & 0	I/4	J/3	V/0
6	0	0	X	0	LOAD & 0	I/4	J/3	V/0
7	0	0	0	0	LOAD & 1	I/6	J/3	V/0
8	0	1	0	0	LOAD & 1	I/6	J/6	V/0
9	0	1	0	0	READ & 0	I/6	J/6	V/0
10	1	1	0	0	READ & 0	I/8	J/6	V/25

END

DATE
FILMED

3-87